Object-Oriented Programming Language

06/04/2020

Lab Assignment 11

1. **(100%)**: A maze class hierarchy includes a base class `MapSite` and three derived classes `Room`, `Wall`, and `Door`. The interfaces for each are:

```
enum Direction {North, South, East, West};
```

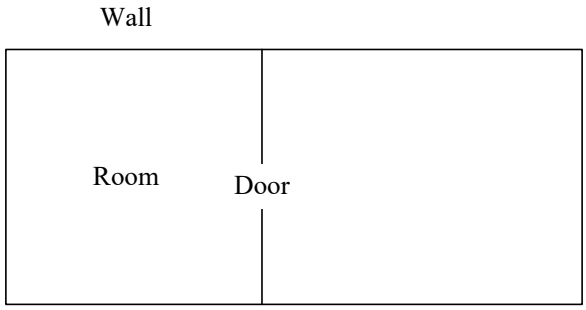
```
class MapSite {
    public:
        virtual void enter() = 0;
};
```

```
class Room : public MapSite {
    public:
        Room(int roomNo);
        MapSite* getSide(Direction) const;
        void setSide(Direction, MapSite*);
        void enter();
  // add your own data members
};
```

```
class Wall : public MapSite {
    public:
        Wall();
        void enter();
};
```

```
class Door : public MapSite {
    public:
        Door(Room* r1= nullptr, Room* r2= nullptr);
        void enter();
  // add your own data members
};
```

Implement this class hierarchy and allow their objects to support creating a maze consisting of two rooms with a door between them and three walls for each room, and adapt the maze creation code using 1.) The Factory Method Pattern, and 2.) The Abstract Factory Pattern, for the `NormalMazeGame` and `BombedMazeGame` (or `NormalMazeFactory` and `BombedMazeFactory`):



And support the following `enter` behavior:
- If you try to enter a wall, you hurt your nose.
- If you try to enter a door, you go into another room.

Note that you will need to add proper data members in addition to the public interfaces for `Room` and `Door` class definition.

Create the two-room maze (stupid maze creation code you will change):

```
Room* r1 = new Room(1);
Room* r2 = new Room(2);
Door* theDoor = new Door(r1, r2);
Wall* w1r1 = new Wall;
Wall* w2r1 = new Wall;
Wall* w3r1 = new Wall;
Wall* w1r2 = new Wall;
Wall* w2r2 = new Wall;
Wall* w3r2 = new Wall;

r1->setSide(North, w1r1);
r1->setSide(East, theDoor);
r1->setSide(South, w2r1);
r1->setSide(West, w3r1);

r2->setSide(North, w1r2);
r2->setSide(East, w2r2);
r2->setSide(South, w3r2);
r2->setSide(West, theDoor);
```

The sample `enter` behavior:

```
r1->getSide(North)->enter();
// cout "Hit the wall and hurt your nose!"

r1->getSide(East)->enter(); // cout "Entering another room"
r1->enter();  // cout "You are in the room No.1."
```

Sample runs (output) look like:

Using `NormalMazeGame/NormalMazeFactory`:

```
Hit the wall and hurt your nose!
Entering another room.
You are in the room No.1.
```

Using `BombedMazeGame/BombedMazeFactory`:

```
Hit the wall and bombed your nose!
Entering another room.
You are in the room No.1.
```